Aulas 1

1. Calcular os nºs de Fibonacci
2. Notação Assimptótica

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{Se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 1

```
Fib(n):
    if n==0 || n==1
      return n
    else
      return Fib(n-1) + Fib(n-2)
```

Esta implementação é eficiente?

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 1

```
Fib(n):
    if n==0 || n==1
      return n
    else
      return Fib(n-1) + Fib(n-2)
```

Esta implementação é eficiente?

$T(n) \rightarrow$ nº de instruções executadas p/ calcular Fib(n)

$$T(n) = \begin{cases} c_0 & \text{se } n = 0 \vee n = 1 \\ c_1 + T(n-1) + T(n-2) & \text{c.c.} \end{cases}$$

$T(n) > Fib(n)$

$Fib(n) \geq (3/2)^n$ for $n \geq 1$

Exemplo 1 - Nºs de Fibonacci

Lema:

$$\forall\, n \geq 11.\quad fib(n) \geq (3/2)^n$$

Prova

Exemplo 1 - Nºs de Fibonacci

Lema:

$$\forall\, n \geq 11.\quad fib(n) \geq \left(\tfrac{3}{2}\right)^n$$

Prova

- A prova faz-se por indução em n
- Caso Base: $fib(11) = 89 \qquad \left(\tfrac{3}{2}\right)^{11} \approx 86.5$

$$fib(11) > 86.5$$

- Caso Indutivo Provar que: $fib(n+1) \geq \left(\tfrac{3}{2}\right)^{n+1}$

$$fib(n+1) = fib(n) + fib(n-1)$$

$$\geq \left(\tfrac{3}{2}\right)^n + \left(\tfrac{3}{2}\right)^{n-1}$$

$$= \left(\tfrac{3}{2}\right)^{n-1} \cdot \left(\tfrac{3}{2} + 1\right)$$

$$= \left(\tfrac{3}{2}\right)^{n-1} \times \left(\tfrac{5}{2}\right)$$

$$\geq \left(\tfrac{3}{2}\right)^{n-1} \times 9/4 = \left(\tfrac{3}{2}\right)^{n+1}$$

observação

$$5/2 = 10/4$$

$$\left(\tfrac{3}{2}\right)^2 = \tfrac{9}{4}$$

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{Se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 2

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{Se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 2

```
Fib(n)
  let c[0..n] be a new array
  c[0] := 0
  c[1] := 1
  for i=2 to n
  |  c[i] := c[i-1] + c[i-2]
  return c[i]
```

Esta implementação é eficiente?

Exemplo 1 - Nos de Fibonacci

$$fib(n) = \begin{cases} n & \text{Se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 2

```
Fib(n)
  let c[0..n] be a new array
  c[0] := 0
  c[1] := 1
  for i=2 to n
  |  c[i] := c[i-1] + c[i-2]
  return c[i]
```

Esta implementação é eficiente?

$$T(n) = c_0 \cdot (n-1) + c_1$$
$$= O(n)$$

$$S(n) = O(n)$$

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 2

```
Fib(n)
  let c[0..n] be a new array
  c[0] := 0
  c[1] := 1
  for i=2 to n
    c[i] := c[i-1] + c[i-2]
  return c[i]
```

Invariante:

$\forall\, 0 \leq k < i.$

$c[k] == fib(k)$

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{Se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 2

```
Fib(n)
  let c[0..n] be a new array
  c[0] := 0
  c[1] := 1
  for i=2 to n
  |  c[i] := c[i-1] + c[i-2]
  return c[i]
```

Esta implementação é eficiente?

$$T(n) = c_0 \cdot (n-1) + c_1$$
$$= O(n)$$

$$S(n) = O(n)$$

Conseguimos melhorar?

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} 1 & \text{se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 3

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{Se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 3

```
Fib(n)
    if n == 0 || n == 1
      return 1
    else
        fib_cur := 1
        fib_prev := 0
        for i=2 to n
          temp := fib_cur
          fib_cur := fib_prev + fib_cur
          fib_prev :=
        return CUR
```

Esta implementação é eficiente?

$T(n) = O(n)$

$S(n) = O(1)$

Exemplo 1 - Nºs de Fibonacci

$$fib(n) = \begin{cases} n & \text{Se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 3

```
Fib(n)
    if n == 0 || n == 1
        return 1
    else
        fib_cur := 1
        fib_prev := 0
        for i = 2 to n
            temp := fib_cur
            fib_cur := fib_prev + fib_cur
            fib_prev :=
        return CUR
```

Esta implementação é eficiente?

$T(n) = O(n)$

$S(n) = O(1)$

Invariante: